Ensuring semantic spatial constraints in virtual environments using UML/OCL

Thanh-Hai Trinh, Ronan Querrec, Pierre de Loor, Pierre Chevaillier

Ensuring Semantic Spatial Constraints in Virtual Environments Using UML/OCL

Thanh-Hai Trinh*, Ronan Querrec†, Pierre De Loor‡, and Pierre Chevaillier§

Université Européenne de Bretagne, France
ENIB, LISyC - EA 3883, CERV, 25 rue Claude Chappe, F-29280 Plouzané, France

Abstract

Spatial objects and relationships between them, compose a spatial model that is the backbone of virtual environments (VEs). However, due to the natural complexity of both spatial objects and spatial information, the modeling of such spatial relationships is still a difficult task. This paper presents a novel approach for representing semantic spatial relations in VEs using the Unified Modeling Language (UML) and the Object Constraint Language (OCL). Our approach first uses the UML class model as a conceptual model for VEs. We then propose a spatial extension of OCL named `VRX-OCL` as a high-level and flexible language to cover multidimensional, manifold, and reference frame-dependent spatial constraints. We mainly focus on two important classes of spatial relations, namely, topological and projective relations that allow nonmetric representation of space. The applicability of our approach is demonstrated in the Virtual Physics Laboratory, a VE for learning physics. Based on the constraints satisfaction, the system is able to visualize abstract spatial information and thus provides educational assistance to the learners.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems —Artificial, augmented, and virtual realities, Evaluation/methodology

Keywords: spatial constraints, spatial languages, semantic virtual environments, domain knowledge visualization

1 Introduction

Spatial relationships between objects provide crucial knowledge about virtual environments (VEs). Particularly, in VEs dedicated to enhancing spatial behaviors such as exploring a space or searching an item in the space, an explicit representation of spatial knowledge is a need [Durlach et al. 2000]. Spatial expressions found in such applications could be "let's move to the statue *in front of* the picture", "the cube is *on* the bench". However, spatial objects populating virtual world are often described in a low-level way within 3D scenes built upon standard formats (e.g., VRML, X3D). As a consequence, spatial relations are implicit and hard to define.

*e-mail:trinh@enib.fr
†e-mail:querrec@enib.fr
‡e-mail:deloor@enib.fr
§e-mail:chevaillier@enib.fr

To overcome these issues, it becomes clear that a high-level representation of spatial objects and notably an explicit description of spatial relationships between them is required. Recent work has focused on defining a semantic model for VEs [Latoschik and Blach 2008]. The main idea is to represent spatial objects not only as animatable 3D shapes but also with their functions, roles, and the relationships between them [Gutierrez et al. 2005]. However, there still lacks a conceptual base and a common language to precisely specify spatial relations and to support the ability to query the spatial knowledge. Difficulties increase when spatial relationships form a constraints network that must be satisfied when users interact with VEs. The problem is not only to express constraints but also to decide whether these objectives are achieved or not. For example, how to compute that a user is near the statue, and consequently, in front of the picture? How to ensure that the user has put the cube on the bench? Especially, spatial constraints can be used as a condition for spatial interactions (e.g., "*Before* activating the laser, verify that the cube is between the lens and the mirror"). To do so, an in-depth formalization of both semantics and computational methods of spatial constraints is needed.

This paper presents a general approach for ensuring semantic spatial constraints in VEs. The novelty relies on the use of the Unified Modeling Language (UML) for the conceptual modeling of VEs and the Object Constraint Language (OCL) for representing spatial constraints. Our motive is that UML is a relevant basis for metamodeling, recognized by both industrial domain and the scientific community. However, this language has not been fully exploited for modeling semantics of VEs. Being an integral part of UML, OCL is a textual language to express and query constraints that diagrams can not cover by themselves [OMG 2006]. Because of the necessity to manage not only logical constraints in the conceptual model but also spatial constraints specialized to 3D dynamic environments, we propose `VRX-OCL`, a *Virtual Reality eXtension of OCL*. Our extension aims at ensuring different semantic constraints in VEs: (1) internal properties of spatial object (e.g., type, value of attributes), (2) spatial relationships among a set of spatial objects, and (3) semantics of spatial object interactions (e.g., pre- and post-condition of spatial tasks). In addition, when associated with adequate spatial reasoning techniques, such a constraint language could be used within an inference system to deduce automatically new knowledge from given one. For example, given "the statue is *in front of* the picture" and "the table is *in front of* the statue", one can conclude that "the table is *in front of* the picture".

By coupling 3D scenes representation with a semantic model, using UML/OCL, our approach gives a novel perspective to define spatial relations between objects that should be satisfied during users's manipulations and to make domain specific knowledge visible to users. Especially, this knowledge can be directly interpreted as instructions displayed textually or graphically, e.g., by visualizing the acceptance areas for the localization of manipulated objects in the 3D environment. VEs for training, learning, or spatial assistance systems are typical use cases that can benefit from our approach.

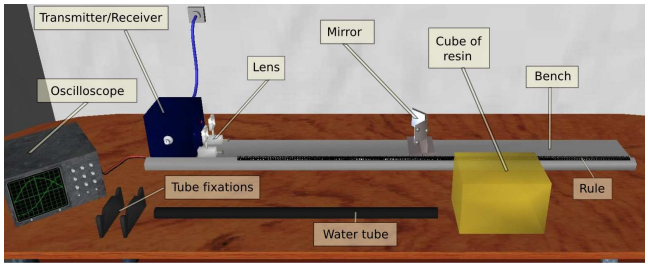
In the next section, we briefly discuss open problems in the modeling of spatial constraints in VEs and related researches. We focus on two important classes of spatial relationships, namely, topological and projective constraints that allow nonmetric representation

of space. We raise the need for a spatial language to model these constraints through examples borrowed from the Virtual Physics Laboratory, a VE for learning physics. Section 3 gives an overview of our approach. Sections 4 and 5 respectively detail how nonmetric spatial constraints are modeled in VRX−OCL. We present both formalization and computational model of topological and projective relations. Section 6 draws our conclusions and outlines future work.

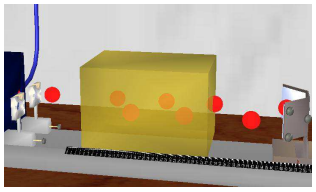
2 Modeling Spatial Constraints in VE - Examples from the Virtual Physics Laboratory

Related researches on modeling spatial constraints in VEs were rare and incomplete. [Pellens et al. 2006] and [Irawati et al. 2006] used spatial ontology to store information about spatial objects and spatial relationships between them. Alternatively, [Kalogerakis et al. 2006] mapped node elements in 3D standards (like VRML, X3D) onto ontology's concepts, relations such as "whole-part" were handled, but not spatial constraints. [Smith and Stuerzlinger 2001] represented spatial objects using scene graph in which each object was constrained to other ones by means of "offer area" and "binding area". [Park et al. 2007] integrated both spatial ontology and scene graph into a framework called NAVER for representing temporal and spatial relations. The main drawback of these approaches is that constraints are directly defined in hierarchical structures, whereas objects populating virtual world are dynamic and often manipulated by users. Smith and Stuerzlinger's model allows constraints to be broken and then re-constrained but they are defined in an ad-hoc manner. Also, only some specific types of constraint are handled.

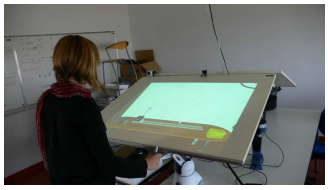
As stated in [Cohn and Hazarika 2001], the most important aspects of space are topology, direction, and distance. Moreover, spatial expressions in natural language are usually described in a qualitative (symbolic) manner (such as "disjoint", "meet", "inside", etc. for topology; "left", "north of" for direction/orientation; "far", "near" for distance). In 3D space, the challenge is that spatial relations are multidimensional (among 3D entities built upon complex shapes), ambiguous (in immersive and multi-user VEs, different points of view might yield different observations of the same relation), and manifold. As pointed out in [Renz and Nebel 2007], spatial relations are often given in the form of constraints that can be unary



(a) The work surface in VPLab (labels were added for explanation).



(b) The effect *Slow motion* is simulated based on spatial constraints.



(c) A user manipulates objects using 6DOF space mouse in desktop mode.

Figure 1: *The Virtual Physics Laboratory (VPLab) for practising physics lab work.*

("the length of the bench is 2 metres"), binary ("the cube should be placed on the bench"), ternary ("the cube should be placed between the lens and the mirror"), or *n*-ary in general. Spatial constraints may relate to metric (distance, angle) or nonmetric information (topology, projection). The later is the subject of this research.

Figure 1(a) illustrates the Virtual Physics Laboratory (VPLab), a VE for learning dedicated to lab work in physics, particularly in optics. One of the exercises learners have to perform is to measure the light speed in different mediums: air, resin, and water. Below is an example of an instruction given to the learner, corresponding to an expected result of a step in the procedure s/he has to follow.

Constraint 1 *To measure the speed of light in resin materials, put the cube of resin on the bench, between the lens and the mirror.*

No metric information (e.g., angle, distance) was used in this example. Such a relation as "on" (meet) stays invariant under topological transformations (i.e., translation, rotation, scaling), so called topological relations. One can also note that the latter constraint is based on the alignment between objects and can be alternatively formulated as "put the cube on the bench such that the mirror is *after* the lens and the cube". Such constraints as "between", "after" are called projective constraints because they are preserved under projective operations that maintain the collinearity of a point set.

Within the VPLab, it is possible to provide some assistance to the learners. For example, it is possible to represent that the velocity of light depends on the material it passes through. This abstract knowledge can be reified by graphically representing light as virtual photons moving at a speed which depends on the object wherein they are located. This assistance is called *Slow motion* (Fig. 1(b)).

Constraint 2 *When Slow motion assistance is active, if a virtual photon is inside an object, its speed is proportional to the refractive index of the material of this object, else its speed is proportional to the refractive index of air.*

In this case, it is necessary to handle in the same expression both object properties and spatial relations. We will see later that the former can be expressed using standard OCL, not the latter. Moreover, user-system interactions in VEs can be carried out either in immersive mode or in desktop mode (see Fig. 1(c)). In the latter case, the user has only a projective view of the 3D scene. Even in immersive mode, sometimes it is useful to give to the user a global view of the environment (e.g, a navigation map in a large-scale environment). These needs are satisfied by means of orthographic views of the environment (i.e., front, top, or side views). The scene in Fig. 1(a) gives an example of such an orthographic projective constraint.

Constraint 3 *From a top view of the work surface, the water tube shall be on the right side of the lens and the mirror.*

In this example, the relative position of the water tube to the lens and the mirror is inverted (leftside instead of rightside) under the opposite view (the bottom-view in this case). Thus, it is necessary for spatial languages to be able to disambiguate spatial constraints.

3 The Proposed Approach

3.1 Building Conceptual Model Using UML

To model spatial objects, we use the UML class model in which every class corresponds to a specific type of spatial objects. Figure 2 illustrates the simplified class diagram representing the 3D scene of the VPLab. For each class, the attributes represent the internal properties of a spatial object (e.g., the length of a bench); the interactions with the spatial object are defined by means of operations. Logical and compositional relations between spatial objects are expressed

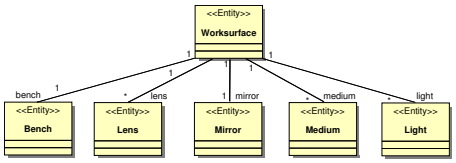


Figure 2: *The simplified UML class diagram representing the work surface of the VPLab.*

via associations and composition links between classes. Interestingly enough, in our conceptual model, even abstract concepts such as light may be modeled by a spatial class, named `Light`, whose instances are considered as photons. We define also some special classes that contain system information (e.g., different camera positions are stored in the `Camera` class).

Thanks to object-oriented approach, a spatial object can inherit attributes, operations, and associations from the parent class. For example, some primitives coming from the underlying 3D library are common between spatial objects, such as rotation, scaling, translation, etc. Furthermore, the modeling of domain-oriented concepts in UML is facilitated by means of *stereotype*, i.e., a new modeling primitive derived from existing ones. In our conceptual model, spatial objects are defined using a new stereotype called `<<Entity>>` that contains the following basic information about spatial object:
- Unique name: is an (string) attribute that uniquely identifies a spatial object. For example, *'transmitterLens'* and *'receiverLens'* are the names of two lenses in the VPLab; *'photon1'*, *'photon2'*, etc. uniquely identify the instances of the `Light` class.
- Position: is a vector representing the coordinates of an object in 3D space. In large-scale environments such as VEs, the position of the object is often defined as its centre of gravity.
- Shapes: refer to the geometric structures of the spatial object. We thus separate conceptual and graphical representations of objects.

Since spatial objects are complex and thus not relevant to manipulate or calculate at the conceptual level, they are often approximated using simpler structures. We hence use in the conceptual model two levels of approximation.
- Semantic points: every spatial object is associated with one *referential point*. Other points may be involved if necessary, e.g., the position, the interaction points, or the markers that facilitate the navigation between objects in VE.
- Convex bounding volume: a spatial object is described by its *axis-aligned bounding box* (AABB). The use of AABB in the conceptual model makes spatial objects easy to compute, cheap to store (only two points are needed for their representation), and fast to test for intersection. Furthermore, this primitive structure provides a convex approximation that is detailed enough to deal with non-metric information such as topology and projection. Although algorithms in the rest of this paper for computing topological and projective models are detailed using AABB, further implementations can be similarly derived for other levels of approximation based on bounding volumes.

As stressed in [Ibáñez and Delgado-Mata 2006], most of this basic information can be automatically annotated. Compared to previous work using ontology as a underlying semantic layer of VE, UML is more expressive and flexible thanks to the concept of stereotype. A further advantage of UML relies on the ability to model operational semantics that is still limited in ontology.

3.2 Extending OCL for Ensuring Spatial Constraints

OCL is a declarative language to specify constraints and expressions on any elements in a UML diagram. It is both a constraint and a query language: an OCL expression returns a value or an object within the system, but the evaluation of expressions does not change the state of the system. Using the UML class model presented in Fig. 2, the following examples illustrate the expressiveness of OCL.

Example 1 The constraint "the length of the bench must be inferior to 2 metres" is formally specified as follows.

```
context Bench inv:
  self.length < 2
```

This constraint is defined in the context of the `Bench` class, `self` represents any instance of this class. It is an invariant that specifies a static property of the attribute `length` of the contextual class.

Example 2 In OCL, the `allInstances()` function returns the collection of all the instances of a class. Universal and existential quantifiers are denoted as `forAll` and `exists`. Logical relations and implication are also supported using logical operators such as `and`, `or`, `xor`, and `implies`. The following expression exemplifies that "different lenses must have different positions".

```
context Lens inv:
Lens.allInstances()->forAll(l1, l2 |
  l1<>l2 implies l1.position<>l2.position)
```

Example 3 To express complex constraints with multi-classes, OCL allows the navigation between classes using associations. Such a cardinality constraint as "a work surface has only one bench but several mediums" is expressed as follows.

```
context Worksurface inv:
self.bench->size()=1 and self.medium->size()>=1
```

In this example, the expression `self.bench` returns a collection of all benches associated to a work surface, while the function `size()` counts the number of elements in a collection.

Example 4 OCL expressions can be used to form a pre- or post-condition of an operation. For instance, the prototype of the constraint "Before activating the laser, verify that the cube is between the lens and the mirror" is as follows.

```
context TransmitterReceiver::activateLaser():Boolean
pre: <a spatial constraint expression>
```

Thus, whenever the operation `activateLaser` on the `TransmitterReceiver` is launched by the user, the `pre` expression is verified to enable/disable this interaction.

Through the above examples, we can see that OCL facilitates the specification of constraints in a formal way. However, invariants in the original OCL are usually local and unary, whereas spatial constraints might be binary, ternary, or *n*-ary in general. Further, as discussed in Sect. 2, in Constraint 3, opposite orthographic viewpoints may yield uncommon observations about the relative position of the water tube (called *primary object*) to the mirror and the lens (called *reference objects*). This is due to the fact that most spatial relations must be given with respect to an implicit or explicit *frame of reference* (FoR) that can be: intrinsic (the relation is given by inner properties of the reference object), extrinsic (the relation is imposed by external factors on the reference object), and deictic (the relation is given by the point of view from which the reference object is seen) [Hernández 1994]. The benefits of this third-person perspective in VEs are also reported in [Salamin et al. 2006].

Addressing these issues, we propose `VRX-OCL` as a spatial extension of OCL. We distinguish two types of spatial expressions: those which are independent of FoR (such as topological constraints, or projective constraints in immersive mode), and those which depend on FoR (projective constraints in orthographic view mode, directional constraints, etc). Thus, generally speaking the syntax of spatial expression in `VRX-OCL` is described as below.

```
SpatialExp::=SpatialExpWithOutFoR|SpatialExpWithFoR
SpatialExpWithOutFoR ::=
```

```
  po '.'  rel_name '(' ro (',' ro)* (',' arg)?  ')'
SpatialExpWithFoR ::=
  SpatialExpWithOutFoR '@' 'viewpoint' '(' obs ')'
```

In this syntax, `po` is the primary object that can be associated with one or more reference objects noted as `ro`. Together with the `@viewpoint` keyword, `obs` is a spatial object that plays the role of an observer from which the relationship is seen. When a spatial expression is associated with an FoR, the constraint evaluation depends on the spatial operator involved to decide whether the direction/position of the observer, the inner properties (e.g., size, internal orientation) or the external factors (e.g., movement direction if the object is being moved) of reference objects must be taken into account. In the case of nonmetric constraints, it is the position of the observer that will be used to unambiguously specify projective constraints under different orthographic views. `arg` is a list of arguments (e.g., projection plane, approximation level, etc.). Finally, `rel_name` stands for the name of the spatial constraint.

4 Modeling Topological Constraints

The two well-known models of topological relation are *RCC-8* [Randell et al. 1992] and *9-intersection* [Egenhofer and Franzosa 1991]. However, the main issue of these models, if applied to VEs, is that they are based on an exact computation of the intersections between objects with sharp boundary (cf. Fig. 3). Meanwhile, in VEs, users manipulate objects using interface devices (cf. Fig. 1(c)) which is a procedure lacking in precision. That makes such a manipulation as *"put the cube meeting the bench"* become difficult if objects involved must have common sharp boundaries.

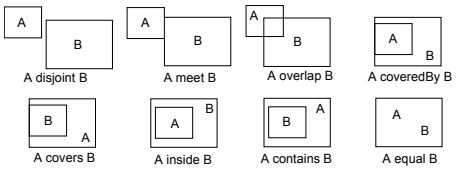


Figure 3: *Illustration in 2D of eight topological relations according to RCC-8 and 9-intersection models.*

To overcome this issue, we propose that spatial objects are represented by AABB whose surfaces have a thickness $\varepsilon > 0$. Considering large-scale VEs, we also suppose that spatial objects are large enough so that every dimension of their AABB is larger than 2ε, so called *thick boundary objects* (TBO). For example, Fig. 4 illustrates the cube whose thick boundary is the buffer zone inside the object, between the red and the blue parts. Similar to the *9-intersection* model, we denote the exterior (the set of points not contained in AABB), the boundary (the set of points contained in the thick surfaces), and the interior (the set of points contained in AABB but not in the thick surfaces) of a TBO A by A^-, A^ε, and A^o respectively. Topological relations between the two TBO A and B are defined by a 3x3 matrix representing the nine intersections between their six parts.

$$I_{TBO}(A,B) = \begin{pmatrix} A^o \cap B^o & A^o \cap B^\varepsilon & A^o \cap B^- \\ A^\varepsilon \cap B^o & A^\varepsilon \cap B^\varepsilon & A^\varepsilon \cap B^- \\ A^- \cap B^o & A^- \cap B^\varepsilon & A^- \cap B^- \end{pmatrix} \quad (1)$$

If one makes an orthographic projection of these TBO to one of the three planes of the Cartesian coordinate system, the result is two *rectangles* with *broad boundary* (BBR) A_{Oth}, B_{Oth} as defined in [Clementini and Di Felice 1997]. Their interiors, boundaries and exteriors are respectively noted as A^o_{Oth}, A^ε_{Oth}, A^-_{Oth} and B^o_{Oth}, B^ε_{Oth}, B^-_{Oth}. Consequently, the intersection matrix between two

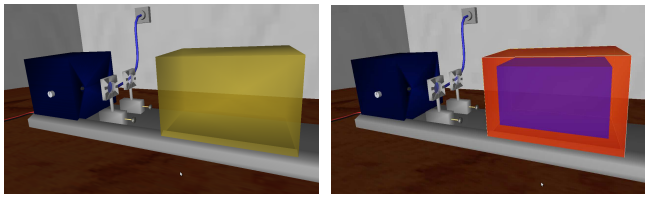


Figure 4: *Visualization of an object without (left figure) and with (right figure) thick boundary.*

BBR is easily defined, called $I_{BBR}(A,B)$.

$$I_{BBR}(A,B) = \begin{pmatrix} A^o_{Oth} \cap B^o_{Oth} & A^o_{Oth} \cap B^\varepsilon_{Oth} & A^o_{Oth} \cap B^-_{Oth} \\ A^\varepsilon_{Oth} \cap B^o_{Oth} & A^\varepsilon_{Oth} \cap B^\varepsilon_{Oth} & A^\varepsilon_{Oth} \cap B^-_{Oth} \\ A^-_{Oth} \cap B^o_{Oth} & A^-_{Oth} \cap B^\varepsilon_{Oth} & A^-_{Oth} \cap B^-_{Oth} \end{pmatrix} \quad (2)$$

Based on the properties of orthographic projections and AABB (two projection rectangles enable the reconstruction of a unique AABB), the following theorem identifies the number of topological relations between two TBO.

Theorem 1 *The number of topological relations between TBO described by the matrix I_{TBO} equals the number of topological relations between BBR described by the matrix I_{BBR}.*

Proof Taking an arbitrary point $p \in A^-$, we have $p \in B^-$ iff $p \in B^-_{Oth}$ in two orthographic projections (easily given by the characteristic of orthographic projection). Thus, $A^- \cap B^- \neq \emptyset$ (sharing a common point) iff $A^-_{Oth} \cap B^-_{Oth} \neq \emptyset$ in two orthographic projections. If we perform the same operation with the other elements in the $I_{TBO}(A,B)$ matrix, it is obvious that each $I_{TBO}(A,B)$ matrix corresponds exactly to an $I_{BBR}(A,B)$ matrix. In other words, the number of topological relations between TBO equals the number of topological relations between BBR.

Let us now consider topological relations between BBR. [Clementini and Di Felice 1997] defined 44 possible relations for regions with broad boundary. However, the drawback is that the topological relations of this model are not symmetric, i.e., `A.meet(B)` does not yield `B.meet(A)`. To us, the thickness ε is chosen by application designers as a global parameter depending on the accuracy of interface devices. The symmetry of the relations is consequently conserved. Since the rectangle is a special case of a region, we apply the following restrictions (the details are left to the readers).

$$\left.\begin{array}{l} A^-_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \\ A^\varepsilon_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \\ A^o_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \end{array}\right\} \Rightarrow \left\{\begin{array}{l} B^-_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \\ B^o_{Oth} \cap B^\varepsilon_{Oth} \neq \emptyset \end{array}\right. \quad (3)$$

$$\left.\begin{array}{l} A^-_{Oth} \cap B^\varepsilon_{Oth} = \emptyset \\ A^\varepsilon_{Oth} \cap B^-_{Oth} = \emptyset \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} A^\varepsilon_{Oth} \cap B^o_{Oth} = \emptyset \\ A^o_{Oth} \cap B^\varepsilon_{Oth} = \emptyset \end{array}\right. \quad (4)$$

Note that these conditions are symmetric with A_{Oth} and B_{Oth}. Such restrictions exclude 27 cases, keeping 17. We further gather them in 8 basic clusters as illustrated in Fig. 5. We however still keep the same numbering and the name of relations as defined in [Clementini and Di Felice 1997] for compatibility and comparison with previous work. As the intersection matrix between the TBO is similar to the BBR, it makes calculating topological relations between TBO in VE a trivial task.

Some further remarks can be made regarding the TBO. For simplicity, we have defined above the thick surfaces as a buffer zone inside a spatial object. They however may be outside, or both inside/outside the object. These cases are applied to simulate other phenomena (e.g., magnetic effects in the VPLab: an object is considered as

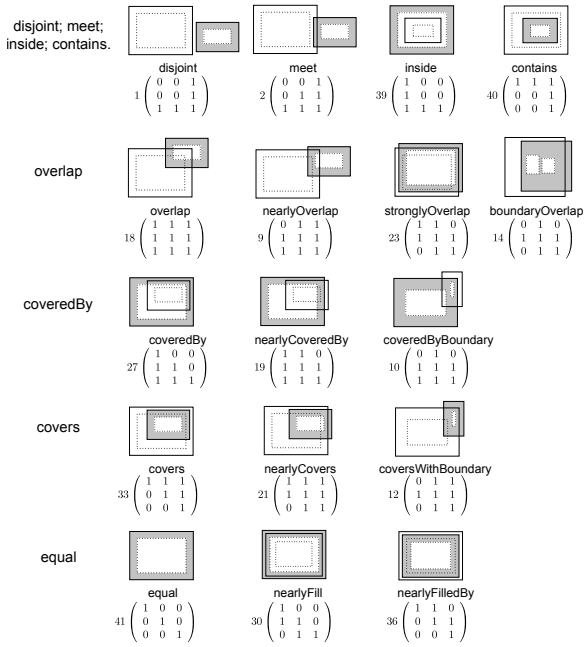


Figure 5: *Illustration in 2D of 17 topological relations between AABB with thick boundary.*

meeting an other object when they are in proximity). In particular, as spatial entities are often large, it seems relevant to suppose that the interior is always bigger than the thickness ε. This restriction eliminates all the configurations in which A^o is inside A^ε (relations numbered 14, 10, and 12), keeping 14 relations.

To sum up, binary topological operators in VRX−OCL take the form of `po.rel_name(ro)`, where `po`, `ro` are respectively the primary and reference objects. `rel_name` is one of the eight basic topological relations: *disjoint*, *meet*, *overlap*, *inside*, *coveredBy*, *contains*, *covers*, *equal*. Using the UML class diagram shown in Fig. 2, the following examples illustrate how topological constraints are described in the VPLab.

Example 5 The part *"the resin cube should be placed on the bench"* in Constraint 1 is expressed in VRX−OCL below.

```
context Worksurface inv:
let resinCube:Medium = self.medium()
        ->select(name='resinCube'),
    bench:Bench = self.bench()
        ->select(name='bench') in
    resinCube->meet(bench)
```

In this example, the `let` expression allows one to define a variable that can be used in the constraint, while the `select` function is used to specify a subset of the collection.

Example 6 The Constraint 2 is expressed as follows.

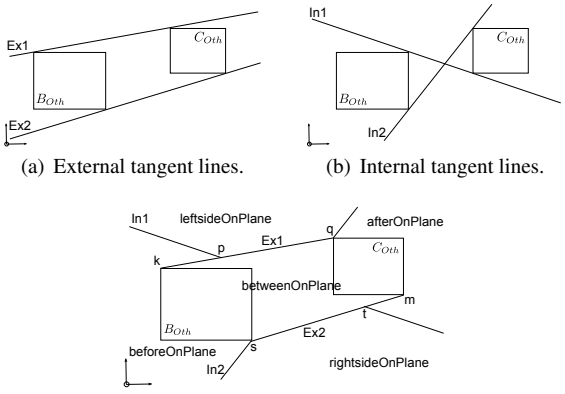
```
context Light::getRefractiveIndex()::Real body:
let r:Medium = Medium.allInstances()
        ->select(name='resinCube'),
    w:Medium = Medium.allInstances()
        ->select(name='waterTube') in
  if self.inside(r) then result = 1.5
  else if self.inside(w) then result = 1.3
  else result = 1 endif endif
```

`getRefractiveIndex()` is a query operation of the `Light` class to get the refractive index of the material in which a photon (`self`) travels. The `result` indicated in the `body` expression is based on topological relations between `self` and the materials.

5 Modeling Projective Constraints

5.1 Ternary Projective Relations

Orthographic View Mode. In order to specify spatial constraints under orthographic views such as Constraint 3, we propose to refine and integrate into VRX−OCL the *5-intersection* model [Clementini and Billen 2006] whose aim is to describe the collinearity among regions. Given three spatial objects represented by their AABB, an orthographic view (i.e., top view, front view, or side view) of a 3D scene results in three rectangles respectively noted as A_{Oth} (primary object), and B_{Oth}, C_{Oth} (reference objects). In the case that the reference objects are disjoint, based on the *5-intersection* model, the relative position of A_{Oth} to B_{Oth} and C_{Oth} is described thanks to the intersections between the external tangent lines (see Fig. 6(a)) and the internal tangent lines (see Fig. 6(b)) which constitute five areas (see Fig. 6(c)). The central area is called `betweenOnPlane` corresponding to the convex hull of B_{Oth} and C_{Oth}. Based on this area, four other areas are defined that allow one to decide whether the primary object is left/right of or before/after the reference objects.



(a) External tangent lines. (b) Internal tangent lines.

(c) Five acceptance areas.

Figure 6: *Ternary projective relations in plan (orthographic) view.*

Nevertheless, as discussed in Sect. 2, different orthographic views may yield different observations of a concrete relation. Thus, we propose that such a spatial relationship must be formalized in the context of a frame of reference as follows.

`po.rel_name(ro1,ro2,plane,collinearity_level)`
`@viewpoint(obs)` where:

- `po`, `ro1`, `ro2` are respectively three spatial objects,
- `rel_name` is the name of the relationship. The five basic relations are `beforeOnPlane`, `betweenOnPlane`, `afterOnPlane`, `rightsideOnPlane`, `leftsideOnPlane`,
- `plane` is the projection plane that is one of the three planes: `XY`, `YZ`, or `XZ`. Each plane corresponds to an orthographic view,
- `collinearity_level` represents different intersection levels between the primary object A_{Oth} and the area indicated by `rel_name`, involving: 1 - A_{Oth} intersects with the acceptance area; 2 - there exists at least one of the semantic points of the primary object A_{Oth} that is in the acceptance area,
- `obs` stands for the viewer from which the relationship is seen. Among the five basic relations described above, only `leftsideOnPlane` and `rightsideOnPlane` relations need a viewer as an explicit frame of reference to disambiguate the relation of A_{Oth} to B_{Oth} and C_{Oth} under different orthographic views.

Let us now consider the semantic of ternary projective operators. Although algorithms for computing ternary projective relations between regions have been discussed in [Clementini and Billen 2006], they however did not allow one to lift ambiguities occurred under different orthographic views. Moreover, since the results of the orthographic projection of AABB are rectangles instead of regions, we decide to build a customized algorithm, which is also used later to compute ternary projective relations in immersive mode.

Algorithm 1 Computing a ternary projective constraint under an orthographic view.
1. Find the internal tangent lines (noted as $In1$, $In2$) and the external tangent lines (noted as $Ex1$, $Ex2$) of B_{Oth} and C_{Oth}.
2. Divide the projection plane into five acceptance areas.
3. Based on the position of the observer, decide whether A_{Oth} (or one of its semantic points) intersects with a particular area from the viewpoint of the observer.

The Step 1 is carried out based on an observation that the external tangent lines connect two vertices of B_{Oth} and C_{Oth} such that all the remaining vertices are in the same half-plane. The internal tangent lines are those that connect two vertices of B_{Oth} and C_{Oth} such that all the remaining vertices B_{Oth} and C_{Oth} are in different half-planes. In Step 2, it is essential to define the convex hull of B_{Oth} and C_{Oth}. We denote p, q, s, t as the intersection points between the external tangent lines $Ex1$, $Ex2$ and the internal tangent lines $In1$, $In2$; k and m are the intersection points of the external tangent lines with B_{Oth} and C_{Oth}. It is obvious that the convex hull of B_{Oth} and C_{Oth} is the union of B_{Oth}, C_{Oth}, and the quadrangle made by k, q, m, s. In Step 3, for the sake of presentation, we only present in this paper the algorithms conceived to decide whether a point belongs to an area. Not lost generality, suppose that the situation in Fig. 6(c) is obtained from a top view projection. In this case, our algorithms are detailed as follows.

Algorithm 2 Point v belongs to the area `betweenOnPlane` if one of the following conditions holds:
(i) v is contained in B_{Oth} or C_{Oth}.
(ii) v is contained in the quadrangle made by k, q, m, s.

Algorithm 3 Point v belongs to the area `leftsideOnPlane` if all the following conditions hold:
(i) v does not belong to the area `betweenOnPlane`. This test is carried out thanks to Algorithm 2.
(ii) v and s are in different half-planes divided by $In1$.
(iii) v and t are in different half-planes divided by $In2$.
(iv) v and s (resp. t) are in different half-planes divided by $Ex1$.

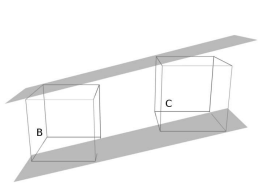
The evaluations for other areas can be derived similarly. In Algorithm 3, if the observer yields a bottom-view of the environment, the result is inverted: the point v that belongs to the area `leftsideOnPlane` from a top-view is in the `rightsideOnPlane` area from a bottom-view, and vice versa. This mechanism obviously enables unambiguous distinction of ternary projective relations observed under orthographic views. Our algorithms are in constant time, compared to that mentioned in [Clementini and Billen 2006] which runs in $O(n\log n)$.

Example 7 Using `VRX-OCL`, the Constraint 3 is expressed as below. For conciseness, only the constraint between the water tube and the transmitter lens and the mirror is illustrated. A similar constraint can be applied to the receiver lens and the mirror.

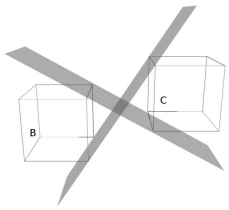
```
context Worksurface inv:
let waterTube:Medium = self.medium()
        ->select(name = 'waterTube'),
    mirror:Mirror = self.mirror()
        ->select(name = 'mirror'),
    tLens:Lens = self.lens()
        ->select(name = 'transmitterLens') in
    waterTube.rightsideOnPlane(tLens, mirror, 'XY', 1)
  @viewpoint(Camera.allInstances()
        ->select(name='TOP_CAMERA'))
```

In this example, we assume that the top-view resulted from the projection of the scene onto the `XY` plane. The evaluation of the constraint depends on `TOP_CAMERA`, an instance of the `Camera` class that contains information about the top-view.

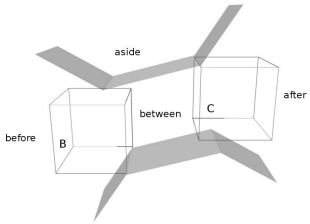
Immersive View Mode. To realize or perceive such spatial operations as Constraints 1 and 2, it is required to embed users into VE in immersive mode. Using haptic devices, this mode enables a more realistic manipulation of spatial entities than desktop mode. To express such constraints in `VRX-OCL`, we are based on the projective model between bodies defined in [Billen and Clementini 2006]. Given two disjoint reference objects B and C, the localization of the primary object A can be retrieved from the intersections between the external tangent planes (see Fig. 7(a)) and the internal tangent planes (see Fig. 7(b)). However, unlike the formal model in which the number of tangent planes are undefined, the use of AABB as a convex approximation of object allows us to argue that there are exactly four external tangent planes and four internal tangent planes of reference objects. Figure 7(c) shows the four acceptance subspaces defined from the intersections between these planes.



(a) Two of four external tangent planes.



(b) Two of four internal tangent planes.

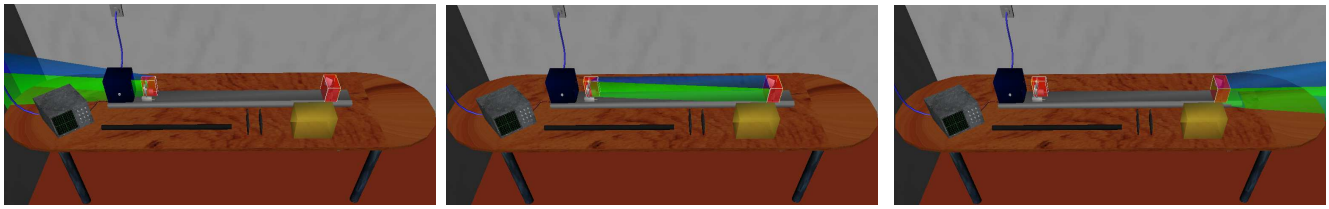


(c) Four sub-spaces made up from the tangent planes.

Figure 7: *Ternary projective relations in immersive view.*

As ternary projective constraints in immersive perceiving mode are inherently independent of the reference frame, we subsequently integrate these relations into `VRX-OCL` as new spatial operators that take the form as follows:
`po.rel_name(ro1, ro2, collinearity_level)` where:
- `po`, `ro1`, `ro2` are respectively three spatial objects,
- `rel_name` is one of the followings: `before`, `between`, `after`, `aside`,
- `collinearity_level` corresponds to the two representation levels of objects in the conceptual model: 1 - the AABB of A intersects with the acceptance sub-space referred by `rel_name`; 2 - there exists at least one of the semantic points of A that is in the acceptance sub-space referred by `rel_name`.

Let us now detail our method to compute ternary projective relations in 3D space that is presently lacking in previous models. The



(a) The zone *before* the lens and the mirror. (b) The zone *between* the lens and the mirror. (c) The zone *after* the lens and the mirror.

Figure 8: *Visualization of the acceptance sub-spaces between the transmitter lens and the mirror in the VPLab.*

most interesting aspect is that the method is based on the reconstitution of the 3D view from orthographic views, as previously described in Theorem 1. For instance, one can note that the primary object A is between the reference objects B and C in immersive view if and only if the image rectangles of A are between the corresponding ones of B and C in two orthographic projections. It leads to the following algorithm for calculating ternary projective relations between volumetric objects.

Algorithm 4 The primary object A is between the reference objects B and C if one of the following conditions holds:
(i) A_{Oth} is `betweenOnPlane` B_{Oth} and C_{Oth} in the two planes XY and XZ.
(ii) A_{Oth} is `betweenOnPlane` B_{Oth} and C_{Oth} in the two planes XY and YZ.
(iii) A_{Oth} is `betweenOnPlane` B_{Oth} and C_{Oth} in the two planes YZ and XZ.

This algorithm is similarly expanded to other sub-spaces. It is obvious that the 3D reconstitution of ternary projective relations from orthographic views is in constant time.

Example 8 In the VPLab, the latter part of Constraint 1 "*put the cube between the lens and the mirror*" is expressed as follows.

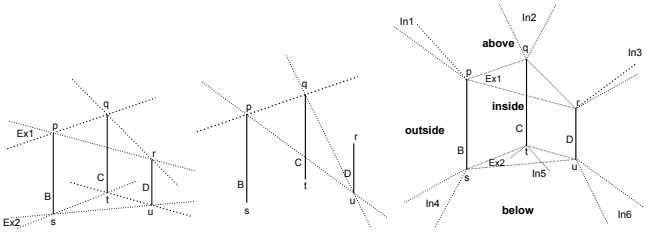
```
context Worksurface inv:
let resinCube:Medium = self.medium()
        ->select(name='resinCube'),
    tLens:Lens = self.lens()
        ->select(name='transmitterLens'),
    mirror:Mirror = self.mirror()
        ->select(name='mirror') in
    resinCube.between(tLens, mirror, 1)
```

A similar constraint can be applied to the receiver lens and the mirror. Within the VPLab, based on the constraints satisfaction, it is possible to visualize abstract spatial relations and thus to help the learners in precisely manipulating spatial objects (see Fig. 8).

5.2 Quaternary Projective Relations

While ternary projective relations are based on the collinearity between three spatial entities, quaternary projective relations deal with the coplanarity among four spatial objects. Given three disjoint reference objects B, C, and D, the intersections between the external tangent planes (see Fig. 9(a)) and the internal tangent planes (see Fig. 9(b)) result in four sub-spaces named *above*, *below*, *inside*, *outside* (see Fig. 9(c), in which full AABBs are not showed). These relations are formalized in `VRX-OCL` as follows.
`po.rel_name(ro1, ro2, ro3,coplanarity_level)` where:
- `po` and `ro1`, `ro2`, `ro3` are respectively the primary and three reference objects,
- `rel_name` is one of the following: `inside`, `outside`, `above`, `below`,
- `coplanarity_level` defines different coplanarity levels between spatial objects: 1 - the AABB of A intersects with the sub-space corresponding to `rel_name`; 2 - there exists at least one of the semantic points of the primary object A that is in the sub-space corresponding to `rel_name`.

In order to compute quaternary projection relations, the first requirement consists of finding internal and external tangent planes. In the context of AABB, we argue that there are two external tangent planes (noted as $Ex1$ and $Ex2$) and six internal tangent planes (respectively noted as $In1$,..,$In6$). Figures 9(a) and 9(b) intuitively show that the external tangent planes connect three vertices from the reference objects such that all the remaining vertices are on the same half-space. In contrast, the internal tangent planes connect three vertices from the reference objects but gather two objects in a same half-space whereas the last one is in another half-space. In addition to the definition of tangent planes, it is needed to define the convex hull of the three reference objects. Let p, q, r and s, t, u be the vertices that respectively make up the two external tangent planes $Ex1$ and $Ex2$, Fig.9(c) shows that the convex hull is built from the union of B, C, D, $Ex1$, $Ex2$, and three planes, termed as *complementary tangent planes*. Informally, an example of a *complementary tangent plane* is the one made up from p, q, s, and t that connects vertices from two of three reference objects such that all remaining vertices are in the same half-space.



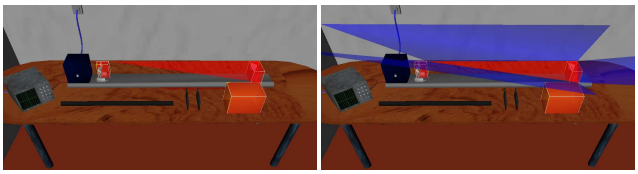
(a) Two external tangent planes. (b) One (of six) internal tangent planes. (c) Four sub-spaces made up from tangent planes.

Figure 9: *Projective relations between four objects.*

Let us now consider the algorithm to test whether a point belongs to a specific sub-space. Due to space limitations, we only present the method for checking the `above` relation which is detailed below.

Algorithm 5 Point v is above the reference objects B, C, and D if all of the following conditions hold:
(i) v is not inside the convex hull of three reference objects.
(ii) v and s, u, t are in different half-spaces divided by $Ex1$.
(iii) v and s (respectively t, u) are in different half-spaces made by $In1$ (respectively $In2$, $In3$).
(iv) v and p (respectively q, r) are in different half-spaces made by $In4$ (respectively $In5$, $In6$).

The integration of these algorithms into `VRX-OCL` as new spatial operators allows one to express and query constraints related to the



(a) The external tangent planes. (b) The zone *above*.

Figure 10: *Visualization of the acceptance sub-spaces between the transmitter lens, the mirror, and the cube.*

coplanarity of a quadruplet of objects. For instance, "Is the oscilloscope coplanar with the lens, the mirror, and the cube? If yes, is the oscilloscope surrounded by three objects?". Figure 10 illustrates some visualization primitives related to quaternary projective relations. The visualization of external tangent planes between the lens, the mirror, and the cube is illustrated in Fig. 10(a). The effect "*Show the acceptance sub-space* `above`" is showed in Fig. 10(b).

6 Conclusions and Future Work

In this paper, we presented a general approach to ensure semantic spatial constraints in VEs. Our first finding was concerned with the use of UML to conceptualize 3D space, thus provided an abstract, implementation independent, and semantic representation of VEs. We then defined `VRX-OCL`, a spatial extension of OCL to formally specify spatial constraints that are inherently manifold and dependent on frames of reference. We presented an in-depth definition of the semantics of nonmetric spatial constraints, i.e., topological and projective constraints. In order to tackle the lack of precision when manipulating 3D objects within VEs, we introduced the concept of *thick boundary objects* and then defined 17 topological relations between them. Our method for formalizing and computing projective relations allowed one to define semantic constraints between three objects (i.e., "before", "between", "after", "leftside", "rightside") and four objects (i.e., "above", "below", "coplanar"), both in orthographic and immersive views. Through the VPLab, we argued that the combination of these two families of spatial constraint offered the ability to define complex constraints without any metric information.

Metric spatial constraints, namely, distance and directional relations will be our future focus. We are particularly interested in qualitative approaches that abstract from metrical details of the physical world and thus provide a spatial representation closer to mental model of humans [Cohn and Hazarika 2001]. Further, we plan to define and incorporate into our model suitable reasoning techniques. The main goal is to enable virtual agents to find out new relationships from existing ones. This research forms part of our project that aims at defining UML/OCL as the basis for a generic language dedicated to the semantic modeling of VEs. As semantic model intensively requires the reification and introspection of the conceptual model, we have been investigating our efforts on the design of a UML-based meta-model that might be used as a semantic reflection layer for VEs.

Acknowledgments. We thank the three anonymous reviewers for their fruitful suggestions, and Christine Baissac for her helpful editing. This work was supported by *La Région Bretagne*, France.

References

BILLEN, R., AND CLEMENTINI, E. 2006. Projective relations in a 3D environment. In *GIScience*, vol. 4197 of *LNCS*, 18–32.

CLEMENTINI, E., AND BILLEN, R. 2006. Modeling and computing ternary projective relations between regions. *IEEE Trans. on Knowl. and Data Eng. 18*, 6 (June), 799–814.

CLEMENTINI, E., AND DI FELICE, P. 1997. Approximate topological relations. *Int'l J. of Approx. Reasoning 16*, 2, 173–204.

COHN, A. G., AND HAZARIKA, S. M. 2001. Qualitative spatial representation and reasoning: An overview. *Fundam. Inf. 46*, 1-2, 1–29.

DURLACH, N., ALLEN, G., DARKEN, R., GARNETT, R., LOOMIS, J., TEMPLEMAN, J., AND WIEGAND, T. 2000. Virtual environments and the enhancement of spatial behavior: Towards a comprehensive research agenda. *Presence: Teleoperators & Virtual Environments 9*, 6, 593–615.

EGENHOFER, M., AND FRANZOSA, R. 1991. Point-set topological spatial relations. *Int'l J. Geogr. Inf. Syst. 5*, 2, 161–174.

GUTIERREZ, M., VEXO, F., AND THALMANN, D. 2005. Semantics-based representation of virtual environments. *Int'l J. of Computer Applications in Technology 23*, 2, 229–238.

HERNÁNDEZ, D. 1994. *Qualitative Representation of Spatial Knowledge*, vol. 804 of *LNAI*. Springer, Berlin.

IBÁÑEZ, J., AND DELGADO-MATA, C. 2006. A basic semantic common level for virtual environments. *IJVR - International Journal of Virtual Reality 5*, 3, 25–32.

IRAWATI, S., CALDERÓN, D., AND KO, H. 2006. Spatial ontology for semantic integration in 3D multimodal interaction framework. In *VRCIA '06: Proc. of the 2006 ACM Int'l Conf. on Virtual Reality Continuum and Its Applications*, 129–135.

KALOGERAKIS, E., CHRISTODOULAKIS, S., AND MOUMOUTZIS, N. 2006. Coupling ontologies with graphics content for knowledge driven visualization. In *VR '06: Proc. of the IEEE conf. on Virtual Reality*, 43–50.

LATOSCHIK, M. E., AND BLACH, R. 2008. Semantic modelling for virtual worlds a novel paradigm for realtime interactive systems? In *VRST '08: Proc. of the 2008 ACM symposium on Virtual reality software and technology*, ACM, NY, USA, 17–20.

OMG, 2006. Unified Modeling Language 2.0 Object Constraint Language Specification, formal/06-05-01.

PARK, C., JIN, T., HIROSEO, M., AND KO, H. 2007. A framework for VR application based on spatial, temporal and semantic relationship. In *ICVR'07: Proc. of the 2nd Int'l Conf. on Virtual reality*, Springer-Verlag, Berlin, Heidelberg, 329–337.

PELLENS, B., KLEINERMANN, F., AND DE TROYER, O. 2006. Intuitively specifying object dynamics in virtual environments using VR-WISE. In *Proc. of VRST '06*, 334–337.

RANDELL, D. A., CUI, Z., AND COHN, A. G. 1992. A spatial logic based on regions and connection. In *Proc. of the Int'l Conf. on Principles of Knowl. Repres. and Reasoning*, 165–176.

RENZ, J., AND NEBEL, B. 2007. Qualitative spatial reasoning using constraint calculi. In *Handbook of Spatial Logics*. Springer Netherlands, 161–215.

SALAMIN, P., THALMANN, D., AND VEXO, F. 2006. The benefits of third-person perspective in virtual and augmented reality? In *Proc. of VRST '06*, ACM, New York, NY, USA, 27–30.

SMITH, G., AND STUERZLINGER, W. 2001. Integration of constraints into a VR environment. In *VRIC '01: Proc. of the Virtual Reality Int'l Conf. 2001*, 103–110. Laval, France.